Raspberry Pi nelle sue manifestazioni più estreme

Pico Chirurgo Allegro: crea il tuo gioco da tavolo

Studio di registrazione domestico con Raspberry Pi 500

Raspberry Pi

Official Magazine
#153 | Maggio 2025

La rivista ufficiale Raspberry Pi tradotta in italiano per RaspberryItaly

P0WER C0D1NG

PROGETTI PRATICI PER LA PROGRAMMAZIONE AVANZATA

CC BY NC SA

Estratto dal numero 153 di Raspberry Pi Official Magazine. Traduzione di *Zzed* e *marcolecce*, revisione testi e impaginazione di Mauro *"Zzed"* Zoia (zzed@raspberryitaly.com), per la comunità italiana Raspberry Pi www.raspberryitaly.com. Distribuito con licenza CC BY-NC-SA 3.0. *Raspberry Pi Official Magazine* is published monthly by Raspberry Pi Ltd., 194 Cambridge Science Park, Milton Road, Cambridge, England, CB4 0AB.

Pico Chirurgo Allegro

Il prolifico maker Kevin voleva ricreare il classico gioco da tavolo Allegro Chirurgo usando il Raspberry Pi Pico. **Rosie Hattersley** ne è entusiasta

Maker
Kevin McAleer
Kevin costruisce robot e condivide i suoi progetti e le sue competenze tramite trasmissioni online settimanali.

kevsrobots.com

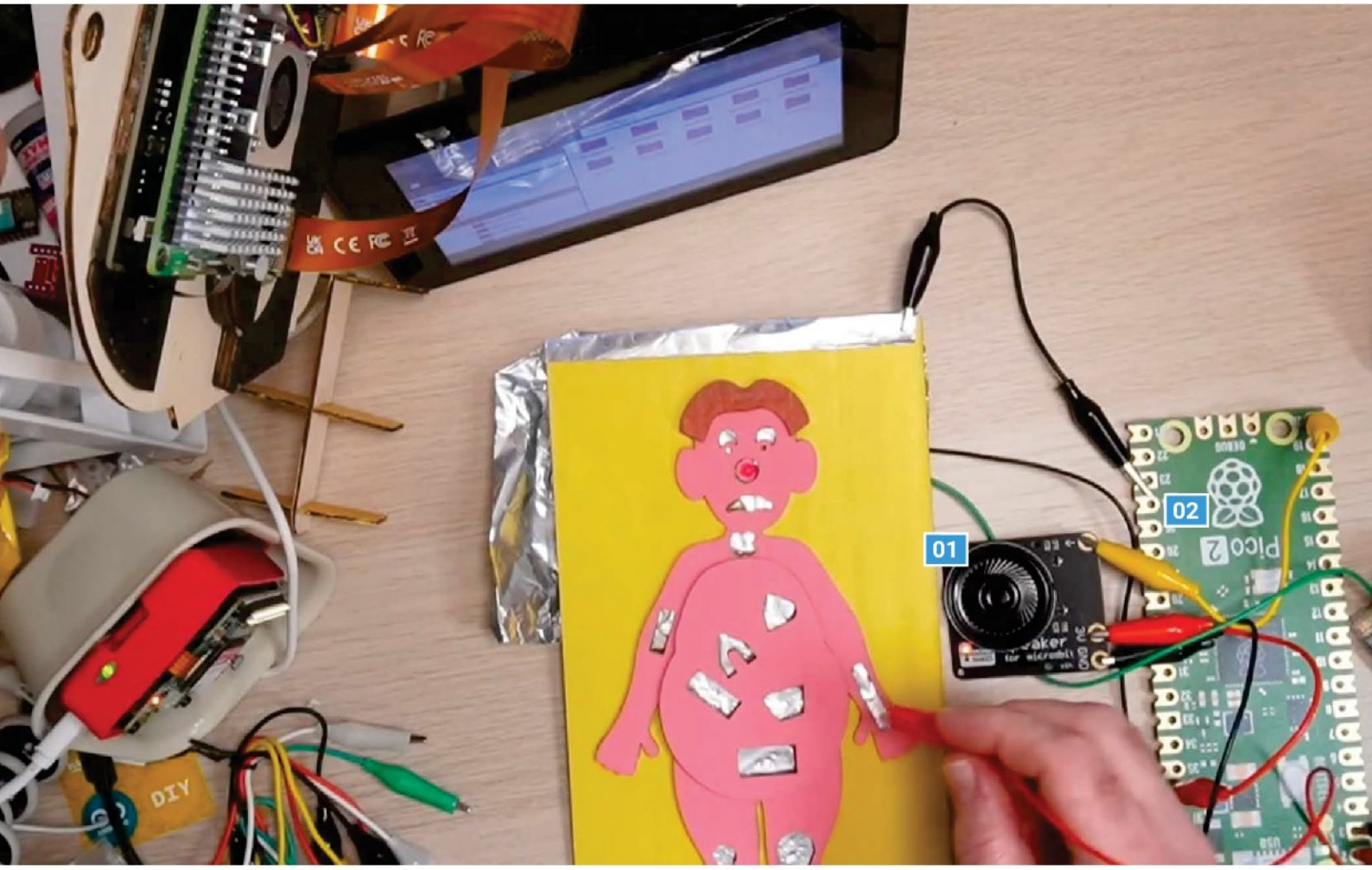
Kevin McAleer è un maestro nel rendere la programmazione divertente, mostrando le possibilità creative attraverso una serie di video e tutorial semplici e spiegati chiaramente, ma memorabili e originali: rpimag.co/kevsyt. Uno dei primissimi progetti di Kevin che abbiamo presentato ha coinvolto un Raspberry Pi e Billy Bass, l'animatronic canterino del momento. Il suo ultimo progetto è una versione fai da te di Allegro Chirurgo, un gioco popolarissimo, amato da chi è cresciuto tra la fine degli anni '70 e gli anni '80. Pico Operation mostra quanto sia facile usare un microcontrollore Raspberry Pi Pico e codice MicroPython per ricreare il gioco da tavolo della Hasbro.

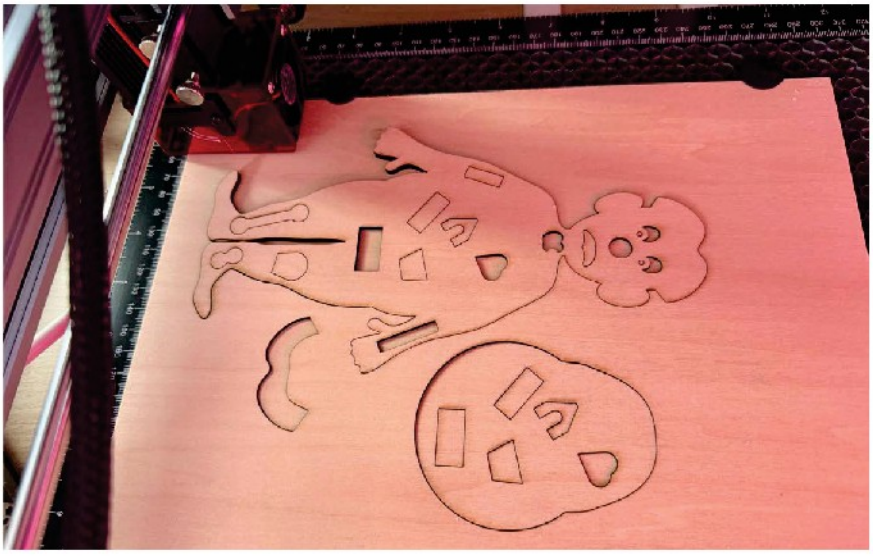
Analizziamolo

Kevin ha un talento nel rendere la programmazione accessibile e ha dato sfoggio della sua inventiva fin da quando ha stampato in 3D il suo case Adafruit Mini Mac con un Raspberry Pi di prima generazione al suo interno. Attribuisce a Raspberry Pi il merito di averlo aiutato a imparare e padroneggiare Linux e Python, il suo linguaggio di programmazione preferito.

Qui, utilizza delle pinze a coccodrillo fissate al Pico Jumbo di Pimoroni – una versione "ridicolmente sovradimensionata" del Raspberry Pi Pico 2 su una PCB compatibile – collegata tramite pin GPIO a un gioco Allegro Chirurgo pazientemente tagliato al laser (**rpimag.co/operation**). Il Jumbo è "perfettamente adatto all'uso con pinze a coccodrillo e quindi non richiede saldature: si può semplicemente sganciare".

> *Pico Chirurgo Allegro ha una serie di risposte melodiose quando una mano poco ferma fa toccare le pinzette con i bordi dei fori e fa suonare il cicalino nel bel mezzo dell'operazione.*





01. Pico Chirurgo Allegro aggiorna il classico gioco buzzer con melodie orecchiabili riprodotte tramite un altoparlante MonkMakes.

02. Un Pimoroni Pico Jumbo controlla tutto, anche se potresti usare un Pico standard

◀ I pezzi di compensato tagliati al laser ricreano il classico paziente di Allegro Chirurgo

Tutto dopo aver giocato e rimettilo sullo scaffale!" È rimasto anche colpito dall'altoparlante MonkMakes che, sebbene progettato per micro:bit, funziona perfettamente con Raspberry Pi.

Il kit include anche un grande naso a LED da 10 mm e una pinza a coccodrillo utilizzata come strumento di rimozione del chirurgo, che "chiude un circuito se tocca il foglio di alluminio sul retro del tavolo operatorio". Il circuito chiuso viene rilevato allo stesso modo della pressione di un pulsante e fa vibrare l'altoparlante e illuminare il naso del paziente.

Ascolta

Pico Chirurgo Allegro ha una serie di risposte melodiose quando una mano tremante fa sì che le pinzette a coccodrillo tocchino i bordi del foro e facciano scattare il cicalino a metà operazione. "Il gioco originale Allegra Chirurgo aveva semplicemente un cicalino collegato alla batteria che attivava un circuito ogni volta che le pinzette toccavano i lati del foro", dice Kevin. "Per questo progetto volevo includere una libreria per riprodurre musica tramite un semplice altoparlante. Ho creato una libreria musicale e ho incluso una breve melodia all'inizio del gioco". Sottolinea che il codice MicroPython per il gioco è lungo solo 45 righe, mentre l'aggiunta della musica lo ha esteso a 107 righe, "quindi è un progetto abbastanza semplice da ricreare".

Era la prima volta che Kevin utilizzava un mini altoparlante amplificato MonkMakes. Dice che "è sorprendentemente forte e conferisce al gioco una dimensione in più, ed è stato molto divertente creare la libreria del lettore musicale". Kevin spiega il processo di creazione musicale nel suo eccellente video a corredo: **rpimag.co/operationyt**.

▲ Kevin ha usato Fusion 360 per ricreare fedelmente l'aspetto e le sensazioni del gioco originale.

Ne ha l'aspetto

L'aspetto più difficile del progetto Pico Chirurgo Allegro è stato ottenere il design perfetto dell'iconico personaggio del paziente. Dopotutto, le famiglie che hanno giocato per oltre cinquant'anni sanno esattamente che aspetto dovrebbe avere. Kevin ha ricreato meticolosamente l'uomo in Fusion 360, usando cerchi e spline per imitare il design originale. Ha poi incollato insieme diversi strati di compensato per dare profondità al personaggio.

Come ha commentato qualcuno nel video di YouTube della realizzazione, "Genio! Adoro come combini il gameplay classico con la tecnologia moderna". Non potremmo essere più d'accordo.

In BREVE

- L'Allegro Chirurgo ha fruttato ad Hasbro 40 milioni di dollari finora
- L'ideatore John Spinello fu pagato 500$ nel 1964
- Il paziente originale si chiama Cavity Sam
- Kevin ha più di 50 progetti con Raspberry Pi all'attivo.
- Non vede l'ora di sentire chi è più veloce!

▼ La scatola del gioco originale, in versione italiana

Body builder

1. Utilizzando i modelli preparati da Kevin (**rpimag.co/operationpico**), ritaglia o taglia al laser i singoli pezzi e dipingili con vernice spray. Applica un foglio di alluminio come base per ogni parte.

2. Utilizza i pin GPIO di un Raspberry Pi Pico 2 o Pimoroni Jumbo per collegare l'elettronica, l'altoparlante MonkMakes alla scheda con le pinze a coccodrillo, insieme a una fonte di alimentazione. Il codice di Operation Pico è disponibile sulla pagina GitHub di Kevin: **rpimag.co/operationgit**.

3. Kevin ha trovato un grosso LED da 10 mm per il naso del paziente, che si attiva ogni volta che il giocatore tocca il foglio di alluminio mentre rimuove una parte del corpo con la pinza a coccodrillo.

P0WER C0D1NG

PORTA LE TUE COMPETENZE DI PROGRAMMAZIONE A UN LIVELLO COMPLETAMENTE DIVERSO. DI ROB MILES

Ci sono due cose che devi fare per diventare un bravo programmatore. La prima è affinare le tue competenze con il tuo linguaggio di programmazione preferito. La seconda è scoprire nuovi linguaggi per ampliare il tuo repertorio. In questo articolo, svilupperemo le nostre competenze in Python mentre creiamo delle luci davvero brillanti. Poi, daremo un'occhiata a un paio di approcci completamente diversi, iniziando da JavaScript e passando a Rust.

SCOPRI NUOVI LINGUAGGI PER AMPLIARE IL TUO REPERTORIO

SVILUPPA LE TUE COMPETENZE IN PYTHON

"ACQUISISCI ABILITÀ CON PYTHON MENTRE GIOCHI CON LUCI E CONTROLLI ACCATTIVANTI"

COSA SERVE

- Raspberry Pi Pico o Pico W (1 o 2)
- Un NeoPixel array 8×8
- Due encoder rotativi

Scopriamo alcune utili funzionalità di Python e usiamole per creare luci colorate. Diamo per scontato che tu conosca un po' di Python (abbastanza per far lampeggiare un LED o accendere dei NeoPixel) e che tu voglia approfondire ulteriormente. Puoi trovare esempi completi di codice qui: **rpimag.co/robmilesgit**. Tutti gli esempi sono in MicroPython e funzionano su qualsiasi versione di Raspberry Pi Pico. Puoi sperimentare utilizzando un pannello NeoPixel e qualsiasi modello di Pico.

PICO LIGHTBOX

Figura 1 mostra il light box. Contiene un array di pixel 8x8 e un Pico. Ci sono anche due encoder rotativi collegati al Raspberry Pi Pico. Ognuno ha un interruttore interno.

Figura 2 mostra lo schema elettrico del dispositivo completo. È possibile utilizzare un pannello di dimensioni diverse, ma potrebbe essere necessario un alimentatore più potente in caso di un numero elevato di pixel.

Se si desidera solo controllare i pixel, è possibile realizzare una versione semplice con un Pico e un pannello display, come mostrato in **Figura 3**. Ora che abbiamo l'hardware, creiamo un po' di software.

◀ **Figura 1:** Progetto finito; i file STL per la stampa 3D del case sono disponibili nel repository GitHub.

TRUCCO VELOCE

Usa LETTERE MAIUSCOLE per denominare le variabili che vuoi utilizzare come valori costanti (immutabili).

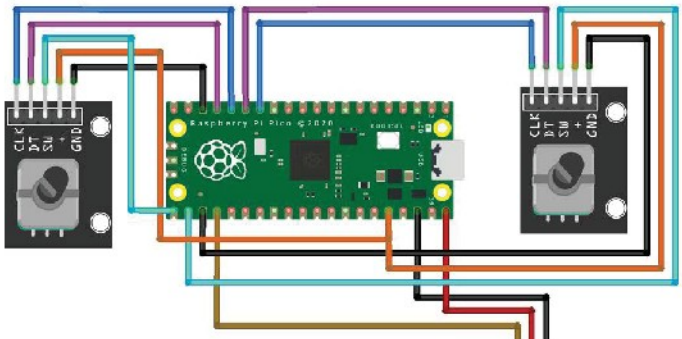
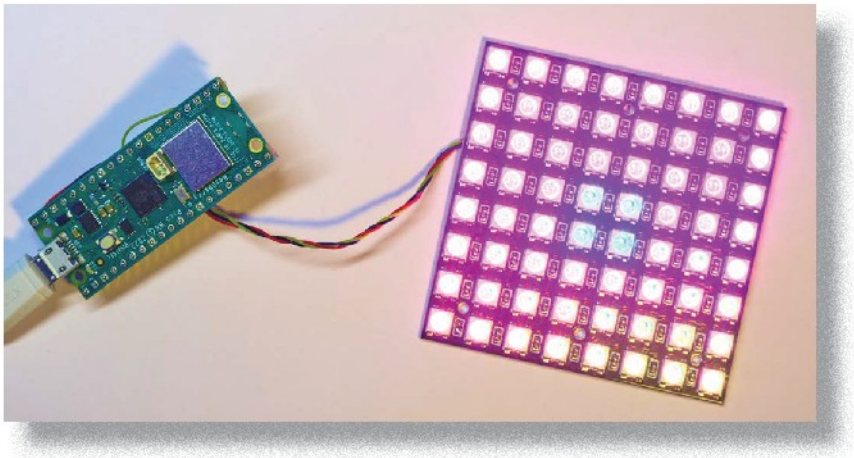


Figura 2: Lo schema elettrico del progetto



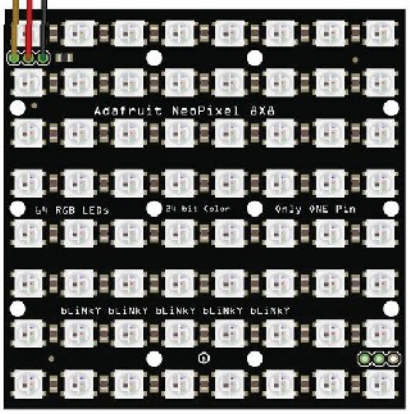


Figura 3: Il pannello è collegato ai pin VSYS, GND e GPIO18 di Pico

UN TOCCO DI CLASSE

Una classe Python è qualcosa che contiene proprietà (variabili di dati) e metodi (funzioni). Creiamo una classe chiamata `Col` per gestire i colori dei pixel nei nostri NeoPixel. Un colore è rappresentato dall'intensità di rosso, verde e blu in esso contenuti. I valori di intensità sono interi compresi tra 0 e 255. Possiamo inserire il colore di un pixel in una tupla Python contenente tre valori:

```
class Col:
    RED = (255, 0, 0)
    YELLOW = (255, 150, 0)
    GREEN = (0, 255, 0)
    CYAN = (0, 255, 255)
    BLUE = (0, 0, 255)
```

Una tupla è come una lista, ma i suoi valori sono immutabili, il che significa che i valori non possono essere modificati una volta creati. Il codice sopra crea una classe chiamata `Col` che definisce alcune proprietà del colore. Ora usiamole per disegnare alcuni pixel.

QUICK ON THE DRAW

```
import machine
import neopixel
```

Le due istruzioni precedenti importano la libreria `machine` che utilizzeremo per interagire con i pin hardware. Viene importata anche la libreria `neopixel`, in modo che il nostro programma possa controllare i NeoPixel.

```
PIXEL_PIN = 18
NUM_PIXELS = 64
```

Queste due istruzioni definiscono il pin del pixel e il numero di pixel nel display. Nel nostro hardware, i pixel sono collegati a GPIO18, quindi il pin del pixel è impostato su 18. Se i pixel sono collegati in modo diverso, è necessario modificare questo valore.

```
np = neopixel.NeoPixel(machine.Pin(PIXEL_PIN),
NUM_PIXELS)
```

Questa istruzione crea un oggetto **NeoPixel** chiamato **np** per controllare i pixel. Ora possiamo usarlo per impostare un pixel di un colore specifico:

```
np[0]=Col.RED
np.write()
```

Il codice sopra imposta il pixel iniziale (in posizione 0) come rosso e poi scrive nuovi colori nei NeoPixel. Il pixel diventerà rosso. Possiamo usare i valori in **Col** ovunque nel nostro programma. Ora mettiamo **Col** al lavoro per noi.

UNA CLASSE PYTHON PUÒ CONTENERE DEI METODI

METODI NELLA NOSTRA FOLLIA

Una classe può contenere metodi. Possiamo trasformare **Col** in un contenitore per il codice di manipolazione del colore, a partire da un metodo che ci permette di creare versioni attenuate dei nostri colori:

```
@staticmethod
def dim(col):
    return (int(col[0]/20),
            int(col[1]/20),
            int(col[2]/20))
```

Il metodo è contrassegnato come **@staticmethod** perché è una parte della classe, non un'istanza. Il metodo **dim** accetta un colore e ne restituisce una versione attenuata. L'attenuazione si ottiene dividendo l'intensità di ciascuna componente (rosso, verde e blu) per 20.

```
np[10]=Col.dim(Col.GREEN)
```

Il codice sopra imposterebbe il pixel numero 10 su un verde tenue. Potremmo aggiungere altre funzioni a **Col** per mescolare i colori o calcolarne di intermedi.

TAVOLOZZE DELIZIOSE

Possiamo anche aggiungere delle "palette" alla classe **Col** in modo da poter creare un set di colori che funzionino bene insieme.

```
 RAINBOW = (RED, ORANGE, YELLOW, GREEN, CYAN,
BLUE, INDIGO, VIOLET)
 FIRE = (RED, ORANGE, YELLOW)
 COOL = (CYAN, BLUE, MAGENTA)
 MONO = (BLACK, GREY, WHITE)
```

Queste sono definite all'interno della classe **Col**. Le useremo più avanti, quando inizieremo a creare luci tremolanti.

COLORI CASUALI

Infine, possiamo aggiungere un metodo a **Col** che sceglierà un colore casuale da una tavolozza:

```
 @staticmethod
 def random_from_palette(palette):
     """Scegli un colore casuale dalla tavolozza
fornita ."""
     if not palette:
         return Col.BLACK  # predefinito
     return palette[random.randint(0,
len(palette) - 1)]
```

Il metodo viene fornito con una tavolozza tra cui scegliere. Utilizza il metodo **randint** della libreria **random** per selezionare un colore casuale da quella tavolozza. Se non viene fornita alcuna tavolozza, il metodo restituirà il nero.

CONTAINER DI CLASSI

Ogni volta che hai un mucchio di valori e piccoli comportamenti che vuoi legare insieme, inserirli in una classe come questa è un'ottima idea. Ora che abbiamo i colori, creiamo delle luci tremolanti.

FAI LAMPEGGIARE UN SACCO DI LUCI CON ASYNCIO

Le luci sfavillanti sono sempre divertenti da realizzare e da guardare. Iniziamo creando una sequenza di colori con un singolo NeoPixel.

```
def flicker_pixel(index, cols, delay):
    while True:
        for col in cols:
            np[index] = col
            np.write()
            time.sleep(delay)
```

Alla funzione **`flicker_pixel`** sopra riportata viene assegnato il numero di pixel da visualizzare (**`index`**), un elenco di valori di colore (**`cols`**) e l'intervallo di tempo tra ogni sfarfallio (**`delay`**). Quindi, scorre ripetutamente i colori, chiamando **`time.sleep`** dopo ogni aggiornamento per mettere in pausa la visualizzazione. Chiameremo la funzione in questo modo:

```
flicker_pixel(0, Col.RAINBOW, 0.5)
```

L'istruzione chiama **`flicker_pixel`** per animare il pixel 0 attraverso la palette arcobaleno con intervalli di mezzo secondo tra ogni colore. La funzione **`flicker_pixel`** non tornerà mai da questa chiamata perché contiene un ciclo **`while`** che non termina mai. Tuttavia, la funzione trascorrerà la maggior parte del tempo inattiva tra i cambi di colore, il che sembra uno spreco.

FAMOLO ASINCRONO

Il framework Python Asyncio consente a un programma di eseguire altre azioni mentre una funzione è "inattiva". Vediamo come possiamo usarlo per creare più pixel lampeggianti che lampeggiano a velocità diverse. La prima cosa che dobbiamo fare è includere Asyncio nel nostro programma:

```
import uasyncio as asyncio
```

Ora possiamo riscrivere la funzione **`flicker_pixel`** per renderla "asincrona".

```
async def flicker_pixel_async(index, cols,
delay):
    while True:
        for col in cols:
            np[index] = col
            np.write()
            await asyncio.sleep(delay)
```

Ci sono solo un paio di modifiche al codice. La prima è che la funzione è ora contrassegnata con la parola chiave **`async`**. La seconda è che la funzione ora chiama **`asyncio.sleep`** per eseguire il ritardo. Vediamo cosa significano queste modifiche.

MONDI INFORMATICI

Normalmente, quando chiami una funzione Python, ottieni il risultato che la funzione restituisce. Tuttavia, quando chiami una funzione contrassegnata come **`async`**, ottieni un riferimento a una "coroutine". Questa è un piccolo oggetto del "mondo informatico" che contiene la funzione e ciò di cui ha bisogno per essere eseguita. I programmi possono funzionare con le coroutine come farebbero con qualsiasi altra variabile.

```
flicker_coroutine = flicker_pixel_async(0,
Col.RAINBOW, 0.5)
```

L'affermazione precedente crea una variabile **`flicker_coroutine`** che si riferisce a una coroutine che contiene una chiamata a **`flicker_pixel_async`**.Più avanti vedremo come avviare l'esecuzione di una coroutine, ma ora diamo un'occhiata all'altra modifica in **`flicker_pixel_async`**: l'uso di **`asyncio.sleep`**.

ATTESE ASINCRONE

La funzione **`time.sleep`** che abbiamo usato nella luce tremolante originale mette in pausa un programma per un tempo specificato. La funzione **`asyncio.sleep`** indica al framework asyncio che la coroutine può andare in modalità sospensione per un po'. Questo significa che asyncio può passare a un'altra coroutine che deve essere eseguita. A un certo punto l'altra coroutine andrà in modalità sospensione, e a quel punto l'esecuzione verrà trasferita altrove. Al termine della sospensione, la coroutine potrà essere nuovamente eseguita. Il framework asyncio è un po' come un giocoliere, che passa da una coroutine all'altra mentre queste sono in esecuzione e in modalità sospensione. Ora possiamo dare vita a questa coroutine, eseguendola.

CODICE IN AZIONE

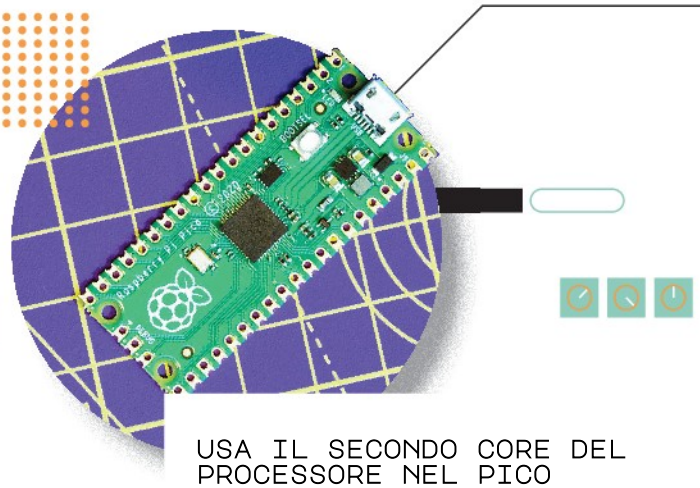
```
asyncio.run(flicker_coroutine)
```

L'istruzione precedente avvierebbe lo sfarfallio di NeoPixel. Il metodo **run** nel framework asyncio accetta una coroutine e la esegue. Finora non sembra molto utile, ma possiamo effettuare diverse chiamate a **run** per avviare più funzioni asincrone.

```
 Led0_coroutine = flicker_pixel_async(0,
Col.RAINBOW, 0.5)
 Led1_coroutine  = flicker_pixel_async(1,
Col.COOL, 0.7)
 asyncio.run (asyncio.gather(Led0_coroutine,
Led1_coroutine))
```

Il codice sopra crea due coroutine per i primi due LED. Quindi le esegue utilizzando la funzione **asyncio.gather**. Quest'ultima accetta più coroutine come parametri e le avvia. Si chiama "gather" perché se le funzioni restituissero risultati (cosa che non fanno quelle relative ai pixel), raccoglierebbe i risultati e li restituirebbe al termine di tutte le funzioni. Ciò che vediamo sono due LED che lampeggiano a velocità diverse utilizzando palette diverse. Stiamo solo scalfendo la superficie di asyncio. È un meccanismo molto potente. Negli esempi di codice di questo articolo, potete trovare un programma che fa lampeggiare 64 LED. C'è anche un esempio che utilizza processi asincroni per controllare il disegno dei pixel e la velocità di lampeggio utilizzando un encoder rotativo. Cose da ricordare sull'utilizzo di asyncio:

- Due processi non vengono mai eseguiti contemporaneamente: questo meccanismo consente semplicemente a una funzione di "restituire" il thread di esecuzione quando non ne ha più bisogno.
- Se una funzione asincrona si blocca e non si ferma mai, bloccherà l'esecuzione di tutte le altre.
- Il modo migliore di condividere i dat per i processi asincroni i è utilizzare un dizionario globale che contenga variabili denominate.



USA IL SECONDO CORE DEL PROCESSORE NEL PICO

Abbiamo visto che possiamo usare asyncio per consentire a un processore di supportare più coroutine, ma queste non vengono eseguite in parallelo. Asyncio, invece, passa costantemente da una coroutine all'altra. Se si desidera eseguire operazioni in parallelo, è necessario più di un processore. Diamo un'occhiata a come funziona e utilizziamo il secondo processore in Raspberry Pi Pico per generare un buon effetto per i nostri pixel

ANIMA E ...CORE

Raspberry Pi Pico ha due core processore. I programmi normalmente vengono eseguiti sul core 0, ma in MicroPython è possibile avviare una funzione in esecuzione sul core 1. La funzione che accede al core 1 si trova nella libreria **_thread**. Un "thread" è un processo che viene eseguito contemporaneamente ad altri. Si noti che un thread non è la stessa cosa di una coroutine. I thread vengono eseguiti contemporaneamente ad altri thread. Le coroutine cooperano per condividere un singolo processore.

```
import _thread
```

L'istruzione precedente importa la libreria **_thread**. Ora abbiamo bisogno di una funzione da eseguire sul core 1:

```
def core1_task():
    while True:
        print("Ciao dal core 1")
        time.sleep(0.6)
```

La funzione sopra scrive ripetutamente "Ciao dal core 1" con un intervallo di 0,6 secondi tra ogni scritta. Ora dobbiamo eseguirla come thread sul core 1:

```
_thread.start_new_thread(core1_task, ())
```

Viene fornita la funzione da eseguire, alla funzione **start_new_thread** seguita dai parametri per la chiamata di funzione. In questo caso, i parametri sono vuoti perché la

funzione core1_task non riceve parametri. A questo punto del programma, la funzione è in esecuzione in un thread sul core 1 e i messaggi iniziano a essere mostrati. Quindi, ora avviamo un altro ciclo in esecuzione sul core 0:

```
while True:
    print("Ciao dal core 2")
    time.sleep(0.5)
```

Questo scrive un messaggio di benvenuto ogni 0,5 secondi. L'output è interessante:

```
Ciao dal core 2
Ciao dal core 1
Ciao dal core 2
Ciao dal core 1Ciao dal core 2
```

I messaggi vengono eseguiti contemporaneamente perché a volte i due thread tentano di scrivere il messaggio contemporaneamente. Questo illustra un problema nell'utilizzo dei thread. Possono finire per litigare per le risorse.

BLOCCO DEL THREAD

Possiamo impedire che i nostri thread si scontrino utilizzando un "blocco" che controlla l'accesso alla funzione **`print`**:

```
print_lock = _thread.allocate_lock()
```

L'istruzione precedente crea un blocco chiamato **`print_lock`**. Solo un thread può avere il blocco in qualsiasi momento.

```
def core1_task():
    while True:
        with print_lock:
            print("Ciao dal core 1")
        time.sleep(0.6)
```

La versione aggiornata di **`core1_task`** utilizza la costruzione Python **`with`** per ottenere il blocco prima di scrivere. Quando l'esecuzione lascia il codice indentato sotto la riga with, il blocco viene rilasciato. Possiamo aggiungere lo stesso meccanismo al codice su core 0:

```
while True:
    with print_lock:
        print("Ciao dal core 2")
    time.sleep(0.5)
```

Un thread attenderà che il blocco diventi disponibile. Se il blocco non diventa mai disponibile, il programma rimarrà bloccato per sempre. Un sistema può contenere più blocchi per proteggere risorse diverse.

BEL RUMORE

L'ultimo esempio (nel repository) utilizza il core 1 per generare una bella visualizzazione del rumore Perlin che viene renderizzata sul display da un processo in esecuzione sul core 0. Funziona perché un thread produce dati e un altro li consuma. Il blocco viene utilizzato solo quando i dati vengono trasferiti tra i processi. Tuttavia, il multiprocessing può essere la fonte di bug orrendi che riguardano la temporizzazione delle interazioni del processore. Cose da ricordare sull'utilizzo del secondo core:

- Esiste un meccanismo che consente a un thread di fare qualcosa di diverso se un blocco è in uso: date un'occhiata al metodo **`lock.acquire(False)`**.
- Se non si utilizza la costruzione **`with`** per ottenere i lock, è necessario prestare molta attenzione alle eccezioni generate che potrebbero impedire il rilascio di un lock e interrompere il programma.
- Entrambi i thread condividono le stesse variabili nel tuo programma.
- Non consentire ai thread di utilizzare contemporaneamente hardware come I2C o connessioni seriali poiché non sono "thread-safe".
- Quando avvii un thread sul core 1, dovrai passare tutti i valori degli elementi necessari per funzionare. Guarda l'esempio di rumore di Perlin per sapere come fare.
- Quando si scrive codice che utilizza il secondo core, potrebbe essere difficile interrompere un programma in esecuzione su di esso.

UN ASSAGGIO DI JAVA SCRIPT

Un programmatore che sa usare un solo linguaggio è come un pilota che sa pilotare un solo tipo di aereo. Non è molto utile. Quindi, tenendo presente questo, diamo un'occhiata a una alternativa a Python: JavaScript.

LA STORIA DI JAVA

Il linguaggio JavaScript è stato sviluppato inizialmente come un modo per rendere più interattive le pagine web. Il codice JavaScript può essere incorporato nelle pagine web per essere eseguito all'interno del browser sul computer di un utente. Tuttavia, la piattaforma Node.js consente di utilizzare JavaScript come linguaggio di uso generale, anche in ambito embedded. Esistono librerie che consentono ai programmi JavaScript, di interagire direttamente con l'hardware. JavaScript è molto simile a Python in quanto consente di ottenere qualcosa di funzionante abbastanza rapidamente. Tuttavia, JavaScript è simile a Python anche perché il codice può essere bizzarro e difficile da capire se non si sa cosa sta succedendo. Diamo un'occhiata a JavaScript attraverso la lente di Python.

INSTALLARE JAVASCRIPT

A differenza di Python, JavaScript non è installato di default su Raspberry Pi OS. Per installarlo, digita quanto segue:

```
sudo apt update
sudo apt install -y nodejs
```

Verifica che funzioni con un semplice file Ciao, Mondo:

```
nano hello.js
```

Inserisci la seguente riga di codice:

```
console.log("Ciao, Mondo!")
```

Uscire e salvare il file (CTRL+O, CTRL+X), quindi eseguire con:

```
node hello.js
```

VARIABILI JAVASCRIPT RILASSATE

Cominciamo con un po' di codice stupido:

```
x = 5
y = "hello"
z = x + y
```

A Python non piace questo codice. Quando lo esegui, ottieni un errore:

```
TypeError: unsupported types for __add__:
'int', 'str'
```

Possiamo decodificare questo messaggio criptico come se Python dicesse: "Mi hai chiesto di aggiungere un intero a una stringa e non so come farlo". Python tiene traccia dei tipi di variabili. Esamina cosa viene inserito in una variabile e ne calcola il tipo. Sa che `x` contiene un numero e `y` contiene una stringa, e non permette di sommarli. Se si vuole davvero aggiungere un numero a una stringa in Python, è necessario prima convertire esplicitamente il numero in una stringa:

```
z = str(x)+y
```

JAVASCRIPT E PYTHON SONO OTTIMI LINGUAGGI

Però, se esegui le istruzioni precedenti in JavaScript, non ottieni alcun errore. E se dai un'occhiata al contenuto della variabile **z**, scopri che contiene la stringa "5hello". Potresti interpretare questo come se JavaScript fosse più intelligente di Python, in quanto sa come aggiungere un numero a una stringa. Potresti anche interpretarlo come se JavaScript fosse più pericoloso di Python, perché potresti aver commesso un errore cercando di aggiungere un numero a una stringa e vorresti sapere quando commetti errori come questi.

Benvenuti nel mondo dei confronti tra linguaggi di programmazione. E proprio come ci sono piloti che affermano che il loro aereo preferito è migliore di quello di chiunque altro, ci sono anche molte persone che affermano che il loro linguaggio preferito è "il migliore". L'autore non ha forti opinioni al riguardo. Gli piace discutere in modo proficuo tanto quanto a chiunque altro, ma pensa che il miglior linguaggio di programmazione sia quello per cui si viene pagati di più. In quest'ottica, JavaScript e Python sono ottimi linguaggi. Approfondiamo un po' di più JavaScript:

PROGRAMMAZIONE JAVASCRIPT

I programmi JavaScript e Python hanno un aspetto leggermente diverso. I programmi JavaScript usano le parentesi graffe per indicare l'inizio e la fine dei blocchi.

```
function add(a, b) {
    let result = a + b;
    return result;
  }
```

Il codice JavaScript sopra crea una funzione **add** che accetta due numeri e ne restituisce la somma. Puoi chiamarla così:

```
x = add(3, 4);
```

Questo imposterebbe il valore di **x** a 7. Tuttavia, la seguente istruzione verrebbe compilata ed eseguita in JavaScript senza errori:

```
x = add();
```

Il valore di **x** verrebbe impostato a NaN, che significa "non un numero". Gli elementi mancanti in una chiamata di funzione vengono trattati come valori impostati al valore "indefinito" in JavaScript. L'aggiunta di due valori indefiniti produce un risultato che non è un numero.

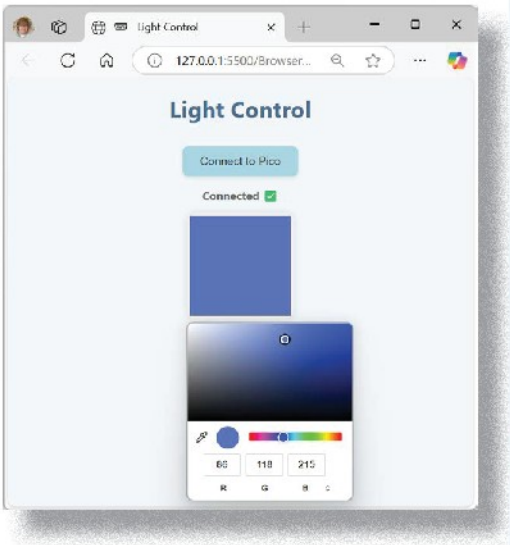
JAVASCRIPT: LO SPETTACOLO DEVE CONTINUARE

JavaScript ha una filosofia del tipo "lo spettacolo deve continuare". Azioni come dividere uno per zero o utilizzare una variabile prima di avervi inserito un valore non causano errori di runtime. Al contrario, JavaScript ha valori integrati nel linguaggio che significano "infinito", "indefinito" o "non un numero". I programmi JavaScript possono (e dovrebbero) testare esplicitamente questi valori durante l'esecuzione per assicurarsi che tutto funzioni correttamente. Questo cambia il modo in cui si scrive il codice quando si usa JavaScript.

Quando si impara un nuovo linguaggio di programmazione, è necessario fare attenzione a non applicare dei presupposti del linguaggio precedente a quello nuovo. E gran parte dell'apprendimento di un nuovo linguaggio di programmazione consiste nel comprendere la filosofia di base del linguaggio. Python è flessibile su alcuni aspetti; ad esempio, JavaScript insiste affinché si dichiarino le variabili prima di utilizzarle nei blocchi, mentre

Figura 5: I segnali di selezione vengono elaborati anche tramite eventi JavaScript

Figura 4: Questa pagina potrebbe essere ospitata sul web in modo che chiunque possa utilizzarla per configurare un dispositivo.



Python no. Tuttavia, JavaScript fa tutto il possibile per garantire che un programma venga eseguito, anche se l'output potrebbe non essere quello desiderato. Quindi, dove si può eseguire JavaScript?

BROWSER JAVASCRIPT

JavaScript in esecuzione in un browser funziona con l'HTML (HyperText Markup Language) che descrive il framework della pagina web all'interno della quale viene eseguito il programma. Questo può essere utile per lo sviluppo embedded perché i browser basati su Google Chrome (incluso Microsoft Edge) contengono una libreria per porte seriali che consente di collegare dispositivi embedded a una porta USB seriale di un computer e quindi inviare comandi.

La **figura 4** mostra una pagina di controllo luci che può essere utilizzata per impostare il colore di un led NeoPixel. Il codice JavaScript nella pagina web invia messaggi di configurazione del colore tramite una connessione seriale al led. Questo contiene un programma Python che imposta il colore e memorizza le informazioni sul colore. Potete trovare questi programmi nel repository di questo articolo.

NODE.JS PER LA VITTORIA

Per eseguire programmi JavaScript su un computer, si utilizza il framework Node.js. Node.js fornisce un ambiente di runtime per i programmi JavaScript. Un Pico non ha potenza sufficiente per eseguire Node.js, ma funzionerà egregiamente anche sul computer Raspberry Pi più economico. Il Node Package Manager (NPM) funziona insieme a Node.js per fornire accesso a una vasta gamma di librerie, come quelle per l'hosting di pagine web, l'interazione con i database e l'interfacciamento hardware, incluso il supporto per GPIO, I2C, SPI e comunicazioni seriali.

La **figura 5** mostra il "telefono rosso". Questo contiene un Raspberry Pi Zero che viene utilizzato per rilevare il sollevamento della cornetta e può simulare una chiamata telefonica, utilizzando la sintesi vocale per inviare messaggi quando si risponde al telefono. Il telefono ospita anche una pagina web che può essere utilizzata per inviare messaggi e far squillare il telefono.

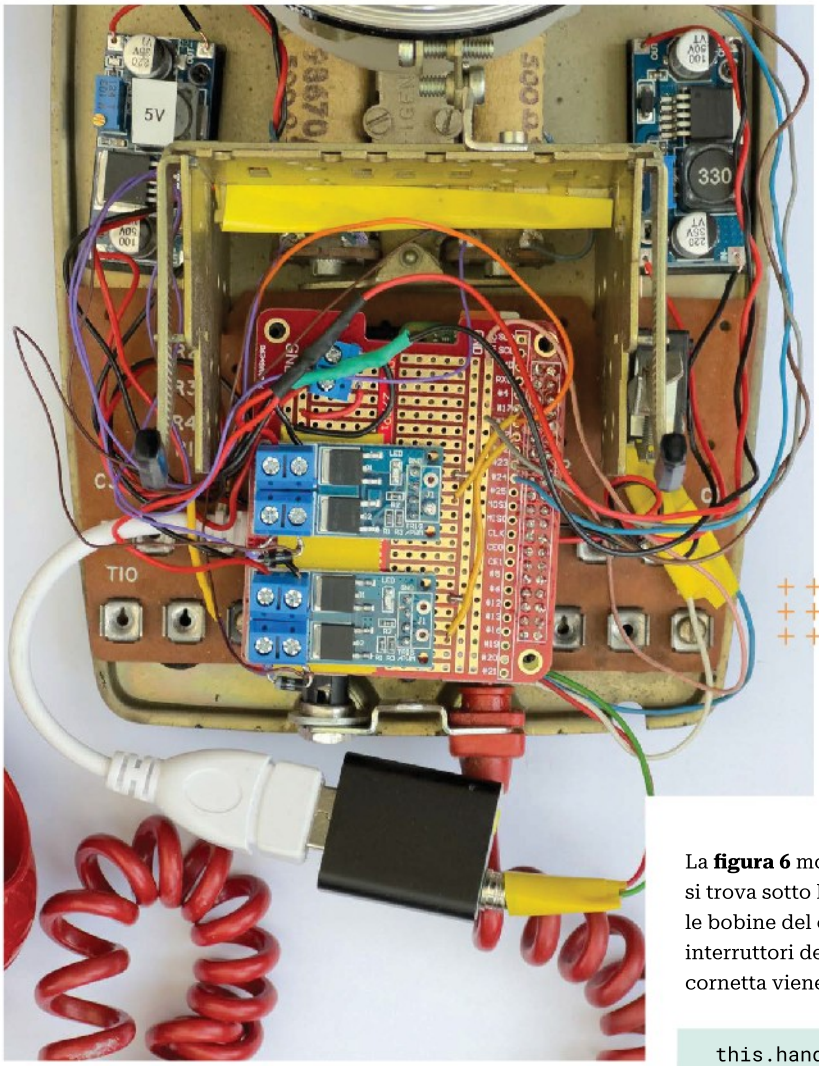


Figura 6: Un'interfaccia audio USB è collegata all'altoparlante del telefono.

La **figura 6** mostra l'interno del telefono. Un Raspberry Pi Zero si trova sotto la scheda di espansione che contiene i driver per le bobine del campanello. Un microinterruttore collegato agli interruttori della cornetta attiva il codice JavaScript quando la cornetta viene sollevata e rimessa giù.

```
this.handsetGPIO = new InGPIO(18, (value) => {
    if(value == 1){
        owner.handsetPickedUp();
    }
    else {
        owner.handsetPutDown();
    }
});
```

L'esempio di codice sopra mostra come il codice JavaScript può essere collegato ai pin di input. Viene chiamato il metodo `handsetPickedup` se l'input cambia stato ed è alto. In caso contrario, viene chiamato il metodo `handsetPutDown`. Puoi trovare tutto il codice JavaScript per il telefono su GitHub qui: **rpimag.co/rpidialphone**.

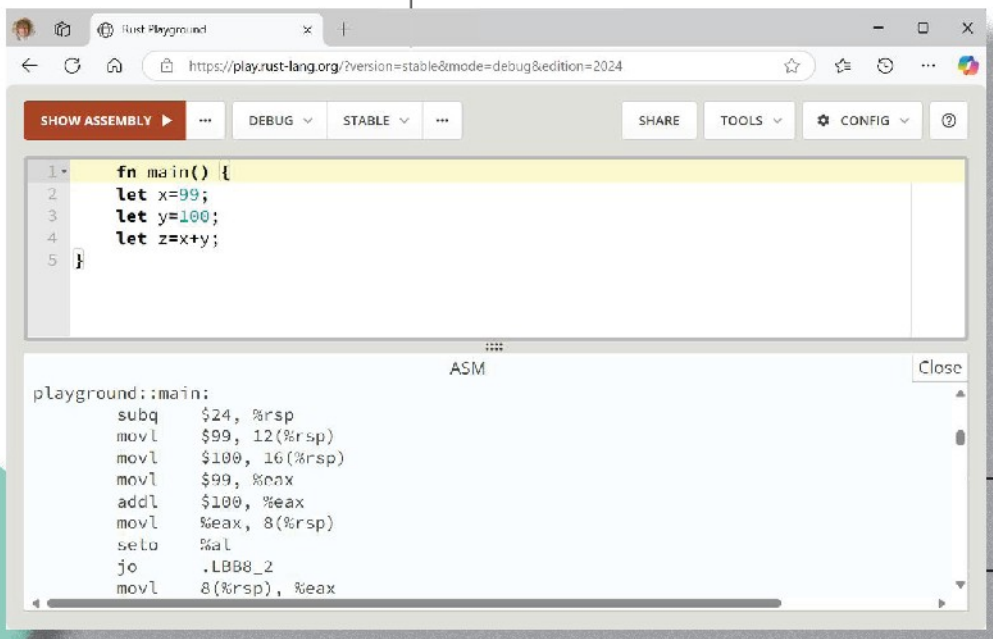
PASSA A RUST

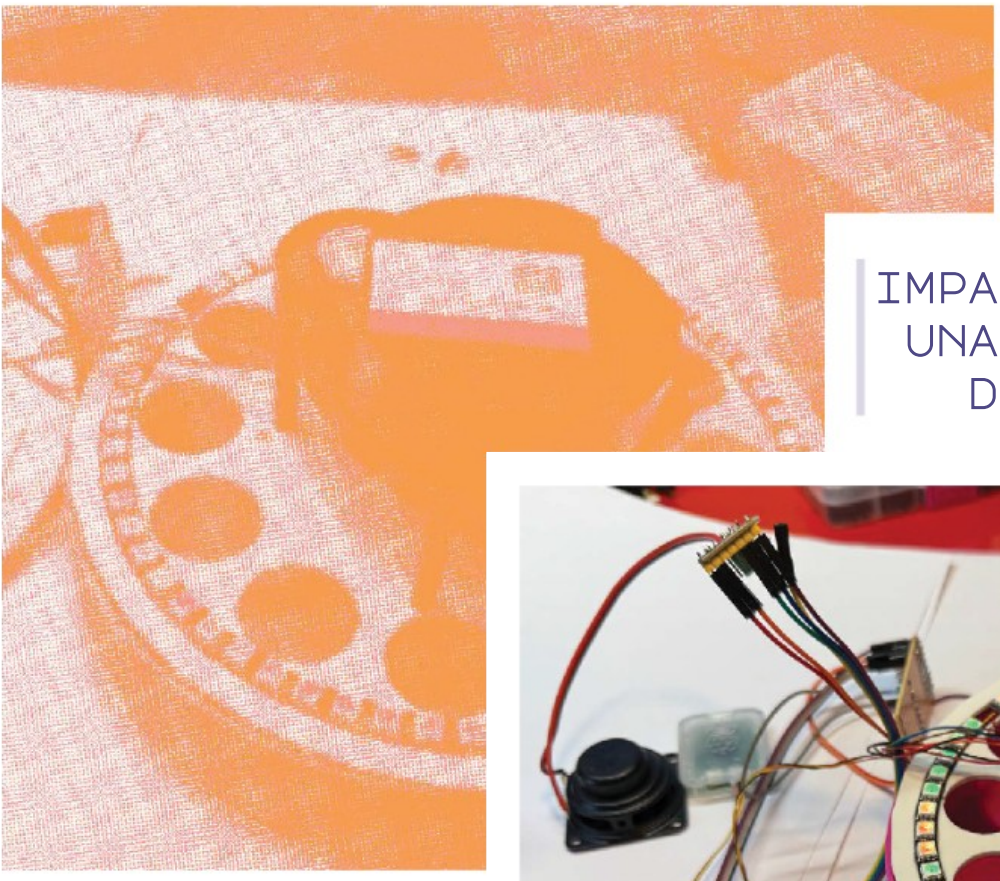
Possiamo concludere questa esplorazione nel mondo della programmazione con uno sguardo al linguaggio Rust. Rust probabilmente non è il primo linguaggio di programmazione che dovresti imparare. Forse non dovrebbe essere nemmeno il secondo. Ma ad un certo punto, ti chiederai se la programmazione sia qualcosa di più e come puoi rendere i tuoi programmi affidabili, autonomi e facili da gestire. Questi sono aspetti che non puoi considerare quando il problema principale è far funzionare il codice, ma una volta che inizi a trovare soluzioni utili, sono aspetti a cui devi prestare attenzione. Rust è stato creato tenendo conto di tutti questi aspetti. Python vuole essere facile da imparare e da usare; JavaScript vuole che ogni programma faccia qualcosa. Rust vuole che ogni programma sia solido, sicuro, prevedibile e costruito per durare. Vediamo come ci riesce.

ESEGUIRE RUST

La **figura 7** mostra un programma Rust che esegue alcune operazioni aritmetiche. Sotto il codice Rust si trovano le istruzioni assembler per quel programma. Quando questo viene eseguito, le istruzioni vengono eseguite direttamente dall'hardware. Ciò significa che i programmi Rust si avviano e vengono eseguiti velocemente. Le istruzioni nei programmi JavaScript e Python vengono decodificate durante l'esecuzione, il che ne rallenta leggermente l'esecuzione. Esistono compilatori Rust per la maggior parte delle piattaforme, inclusi Pico e Pico W (ma non per i dispositivi Pico 2 al momento della scrittura).

Figura 7: Puoi provare Rust su play.rust-lang.org





IMPARARE RUST NON È UNA STRADA FACILE DA PERCORRERE

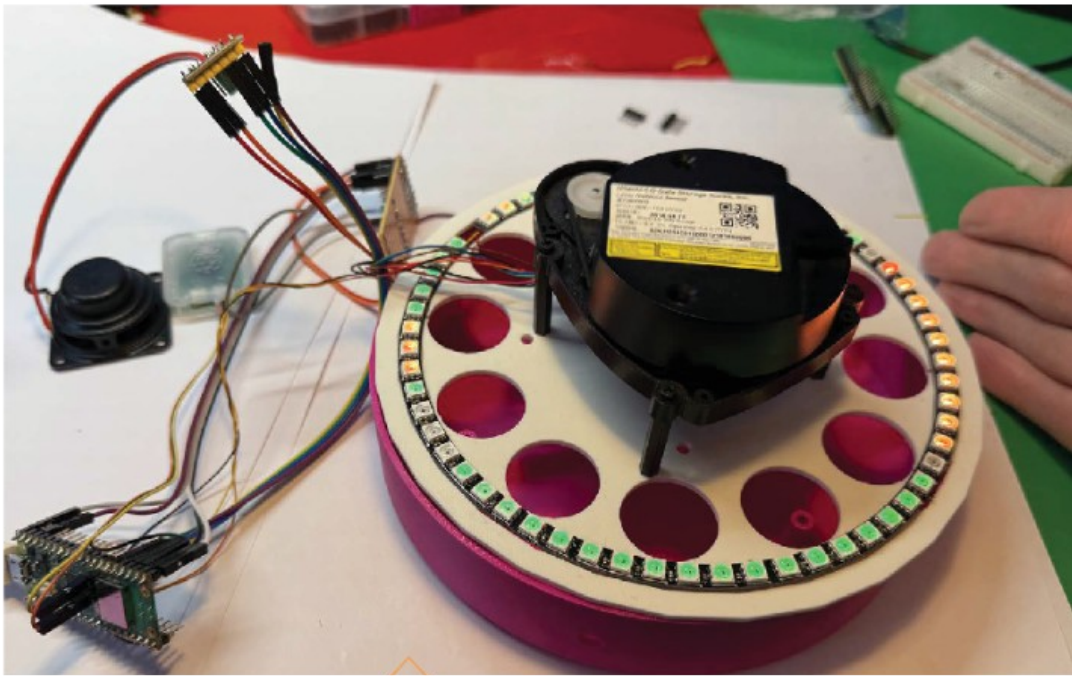


Figura 8: Puoi trovare il progetto su **rpimag.co/rustylidar**

STATICO SULLA LINEA

I programmi Rust sono "collegati staticamente", il che significa che tutto ciò che è necessario per l'esecuzione viene determinato al momento della compilazione del programma. Anche i dati devono essere allocati e gestiti staticamente, in modo che non vi sia alcuna possibilità che un programma Rust esaurisca la memoria o si fermi mentre un garbage collector esegue il processo per liberare spazio. Rust comprende anche un sistema di compilazione che consente di specificare e gestire le versioni dei componenti in una soluzione. Se si stanno creando sistemi critici per un'implementazione su larga scala, questo è molto importante: impedisce al codice di fallire a causa di modifiche alle librerie da cui dipendono.

DIVERTITI CON RUST

La **figura 8** mostra un sensore lidar rotante sopra un anello di NeoPixel. I pixel si illuminano di rosso quando vengono rilevati oggetti vicino al sensore. La fase successiva del progetto consiste nell'aggiungere un'uscita audio in modo da poter creare un gioco di "avvicinamento furtivo" con il dispositivo al centro della stanza. Il Pico elabora il flusso seriale ad alta velocità proveniente dal sensore lidar e poi pilota i NeoPixel. Tutto il software è scritto in Rust, un linguaggio all'altezza della gestione dei dati in arrivo. L'autore si sta divertendo a imparare il linguaggio: è stato progettato per infondere un approccio professionale a chi lo utilizza. Imparare Rust non è facile, ma ne vale la pena.

Parte 03

Realizza uno studio di registrazione domestico con Raspberry Pi 500

Come installare e configurare il software necessario per una produzione audio di alta qualità

Maker

K.G. Orphanides

K.G. sostiene che accordare in Do# e usare gli arpeggiatori sia barare con la musica, e lo fa con grande entusiasmo.

twoot.space/@owlbear

Raspberry Pi OS, come la maggior parte delle altre distribuzioni Linux, offre un'ampia gamma di software di produzione audio di qualità alquanto variabile. Esamineremo le opzioni disponibili per l'audio. Useremo JACK2 per l'audio a bassa latenza insieme a pipewire per l'audio di sistema, con un livello di compatibilità pipewire-pulse per il routing tra JACK e le applicazioni desktop standard.

Per prima cosa, apri un terminale e installa il software audio:

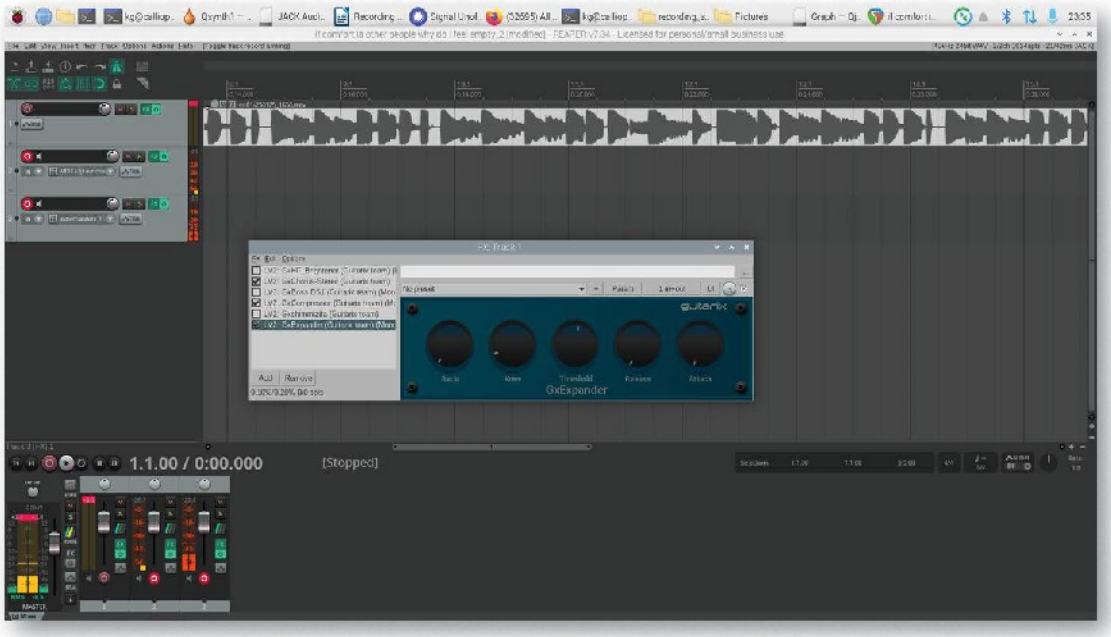
```
$ sudo apt install jackd2 qjackctl
pipewire-pulse qsynth guitarix
```

Quando fatto, digita:

```
$ qjackctl
```

Fai clic su "Setup" e, nella scheda "Settings", assicurati che il driver sia impostato su ALSA e che il driver MIDI sia impostato su "raw". Fai clic su "Advanced" e assicurati che la tua interfaccia audio sia impostata come dispositivo di output e, se disponi di una tastiera MIDI USB stand-alone, che sia impostata come dispositivo di input. Infine, fai clic sulla scheda "Options" e inserisci quanto segue in "Execute script after startup":

```
pacmd set-default-sink jack_out && pacmd
set-default-source jack_in
```



Siamo fan della eccellente interfaccia FX di Reaper per l'audio PCM e delle opzioni di input MIDI.

Imposta la tua interfaccia audio

La maggior parte delle interfacce audio USB è class-compliant, quindi funziona su Linux, ma potresti riscontrare problemi con i dispositivi che si basano su software proprietario per la configurazione o l'aggiornamento del firmware. Abbiamo scelto una Focusrite 2i2 per la configurazione del nostro studio perché offre il pieno supporto Linux grazie alla collaborazione tra Focusrite e la comunità di sviluppo del kernel Linux, e in particolare al duro lavoro dello sviluppatore di driver Geoffrey Bennett. Assicurati di avere una versione compatibile del kernel: il supporto per l'interfaccia audio Focusrite è integrato nel kernel Linux 6.8 e versioni successive. Al momento della stesura di questo articolo, il kernel predefinito per Raspberry Pi era 6.6.74+rpt-rpi-2712; questo non include il modulo per la 2i2, ma fortunatamente il kernel di test (6.12) sì. Il primo passo da compiere è controllare la versione del kernel. Al momento della lettura di questo articolo, il kernel predefinito potrebbe essere stato aggiornato alla versione 6.12. Scopriamolo. Apri un terminale e digita:

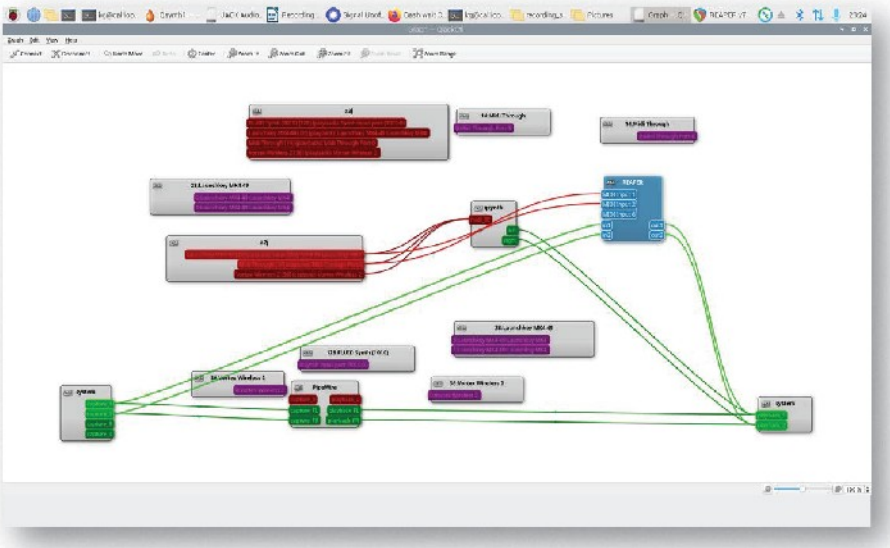
```
$ uname -a
```

Dovrai trovare effetti digitali creati per l'architettura aarch64 o compilarli da solo

audio_start.sh

> Linguaggio: **Bash**

```
001. #!/bin/bash
002.
003. #Esegui tutti i processi richiesti per il driver JACK, le
     connessioni MIDI e carica patch del sintetizzatore SF2
004.
005. jackd -d alsa&
006. a2jmidid -e&
007. qjackctl&
008. qsynth&
009. ;
```



▲ Il grafico delle connessioni di QjackCtl sembra disordinato, ma rende la connessione degli ingressi MIDI particolarmente semplice.

Se segnala una versione del kernel inferiore alla 6.8, aggiorna al nuovo kernel sperimentale:

```
$ sudo rpi-update
$ reboot
```

Installa l'interfaccia di controllo Focusrite

```
$ git clone https://github.com/geoffreybennett/
alsa-scarlett-gui.git
$ cd /home/kg/Software/alsa-scarlett-gui-0.4.0
$ cd alsa-scarlett-gui-0.4.0/
$ sudo apt -y install git make gcc libgtk-4-dev
libasound2-dev libssl-dev
$ cd src
$ make -j4
```

Prova per assicurarti che funzioni:

```
$ ./alsa-scarlett-gui
```

Quindi installalo in tutto il sistema:

```
$ sudo make install
```

L'interfaccia grafica della Focusrite Scarlett dovrebbe essere aggiunta automaticamente alla sezione Altro del menu principale. Puoi usarla per aggiornare il firmware della tua Scarlett, collegare e attivare vari ingressi, regolare il guadagno e altro ancora.

Scelta del microfono

Utilizziamo un microfono a condensatore Blue Yeti Pro collegato tramite XLR, anziché tramite la sua interfaccia audio USB integrata. Sebbene sia possibile utilizzare più interfacce audio, è più semplice gestire la latenza e il routing audio con un singolo dispositivo. Se si utilizza un microfono dinamico, come il nostro Shure SM-58, è consigliabile un preamplificatore, come il pratico FetHead in linea di Triton Audio (**rpimag.co/fethead**). Questo si collega tra il microfono e l'interfaccia audio e offre un boost di 27 dB, il che significa che non sarà necessario aumentare il guadagno del dispositivo audio USB. Sono disponibili anche preamplificatori in linea più economici con un boost leggermente inferiore, così come molti altri più costosi. Questo è utile perché molte interfacce audio USB raggiungono la soglia segnale/rumore a circa il 50% di guadagno, che può causare un fruscio nelle registrazioni.

La tua tastiera MIDI

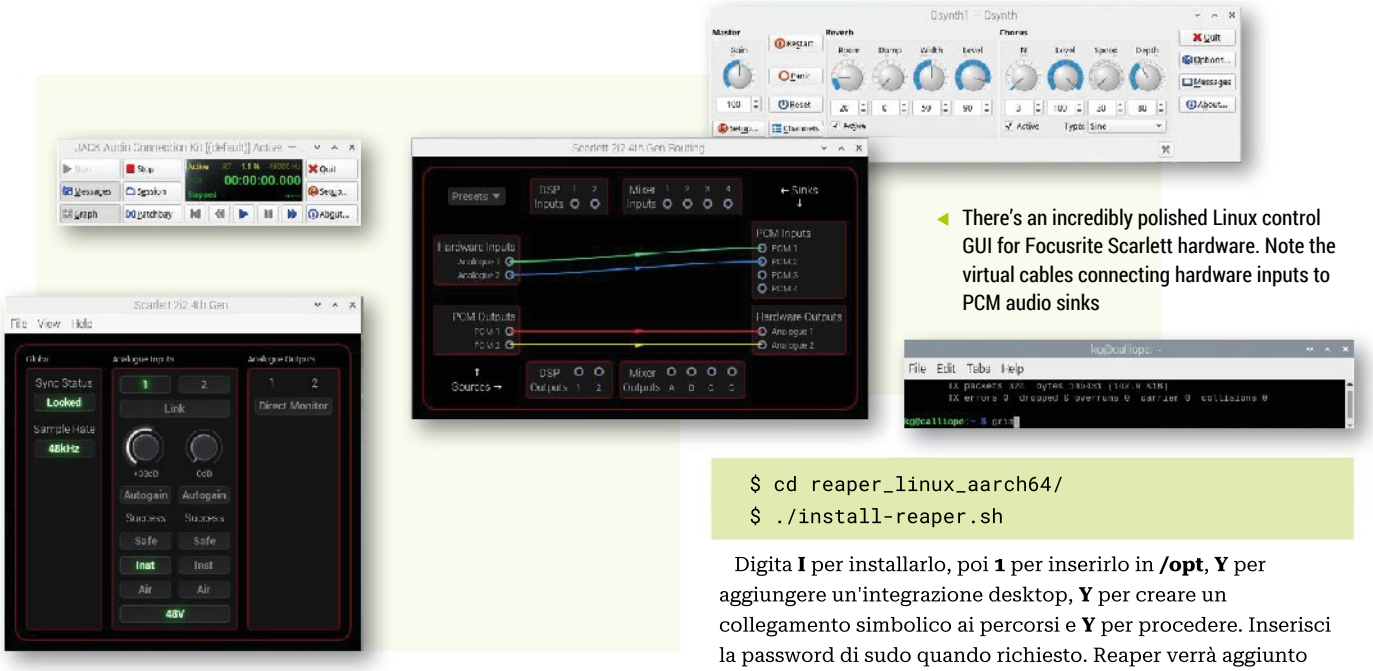
Utilizziamo diversi strumenti MIDI, tra cui una tastiera Novation LaunchKey 49 e una keytar Alesis Vortex Wireless 2. Colleghiamo le tastiere USB al Raspberry Pi direttamente o tramite un hub USB alimentato: gli hub non alimentati non forniscono energia sufficiente.

I driver possono essere un problema su Linux, ma i dispositivi Novation più recenti possono essere gestiti e aggiornati con un browser basato su Chromium tramite un'app web all'indirizzo **rpimag.co/novation**. Questa è una funzionalità relativamente insolita ancora, ma altri marchi che offrono aggiornamenti firmware tramite app web includono PirateMIDI, Morningstar e PandaMIDI. La nostra Alesis non richiede alcuna gestione del firmware, e nemmeno le vecchie tastiere MIDI collegate tramite adattatori USB MIDI.

Una volta configurata la tastiera, il modo più semplice per testarla è tramite Qsynth, un'interfaccia utente grafica per Fluidsynth. Viene fornito con un set di voci General MIDI accurate e di alta qualità, ma puoi usarlo per caricare qualsiasi file SoundFont (SF2), come l'opl3-soundfont, che troverai nei repository.

Instant FX rack

Guitarix permette di combinare pedali digitali per chitarra e altri effetti per ottenere l'equivalente di centinaia di sterline di pedali e strumenti come un accordatore per chitarra. Questi effetti possono essere utilizzati anche nella maggior parte delle DAW, e alcuni sono particolarmente adatti all'integrazione con l'una o con l'altra, come una gamma di effetti LV2 specifici per Ardour. È possibile utilizzare questi effetti, così come gli effetti LADSPA, con Guitarix, mentre le DAW supportano anche i plug-in VST3 e CLAP, tra gli altri.



There's an incredibly polished Linux control GUI for Focusrite Scarlett hardware. Note the virtual cables connecting hardware inputs to PCM audio sinks

Quando scarichi effetti digitali, dovrai trovare file creati per l'architettura aarch64 o compilarli tu stesso. Fortunatamente, molti sono già disponibili nei repository. Alcuni effetti utili verranno installati insieme a Guitarix per iniziare.

Impostare una DAW

Una workstation audio digitale (DAW) è un ambiente completo per la registrazione e la produzione musicale utilizzando strumenti MIDI o analogici dal vivo collegati tramite l'interfaccia audio. Reaper, Ardour, LLMS simile a FruityLoops e il leggero e nostro preferito Qtractor sono strumenti validi, così come Audacity e le sue versioni dedicate all'audio PCM.

Stiamo spingendo al limite le potenzialità di Raspberry Pi 500 per quanto riguarda alcune di queste applicazioni. La maggior parte dei registratori PCM, con la notevole eccezione di Qtractor, ha fatto fatica a mantenere la visualizzazione della forma d'onda a tempo con la registrazione audio, sebbene la cattura audio risultante fosse perfetta.

Anche alcune voci ed effetti di synth ad alto consumo di processore hanno influito sulle prestazioni della DAW. La maggior parte di questi strumenti può essere installata direttamente dalla riga di comando:

```
$ sudo apt install lmms audacity ardour qtractor
```

Reaper è un software commerciale, ma ha un prezzo molto ragionevole di 60 $ (53 €) ed è altamente personalizzabile, con un'interfaccia chiara, numerose funzionalità e una generosa prova gratuita.

Per installare Reaper, visita **rpimag.co/reaperfm** e scarica l'ultima versione Linux aarch64 del software, quindi estraila. Apri un terminale e vai con **cd** alla directory dove lo hai estratto:

```
$ cd reaper_linux_aarch64/
$ ./install-reaper.sh
```

Digita **I** per installarlo, poi **1** per inserirlo in **/opt**, **Y** per aggiungere un'integrazione desktop, **Y** per creare un collegamento simbolico ai percorsi e **Y** per procedere. Inserisci la password di sudo quando richiesto. Reaper verrà aggiunto alla sezione Audio e Video del menu di Raspberry Pi OS.

Realizzare le connessioni

Una volta avviata l'interfaccia di controllo di JACK, l'interfaccia grafica di Focusrite Scarlett e il software musicale che preferisci, molto probabilmente scoprirai di non sentire effettivamente il suono dove ti aspetti. Sia l'interfaccia grafica di Scarlett che l'interfaccia grafica di JACK richiedono di stabilire connessioni tra i dispositivi per assicurarsi che comunichino correttamente tra loro, come mostrato nei nostri screenshot.

Ottenere effetti aarch64 e plug-in per sintetizzatore

Per alcuni validi indici di plug-in audio, visita **rpimag.co/zynthian**, **rpimag.co/lv2plugins** e **rpimag.co/kxplugins**. KXstudio offre anche un repository di software e plug-in su **rpimag.co/kxrepos**.

Numerosi plug-in, inclusi molti di quelli elencati nei siti qui sopra menzionati, sono disponibili nel repository di Raspberry Pi OS. Utilizzate l'interfaccia grafica "Aggiungi/Rimuovi software" per cercare plug-in VST, LADSPA, LV2, CALF e altri ancora.

Oppure, per iniziare, apri un terminale e digita:

```
$ sudo apt install calf-plugins amb-plugins
ardour-lv2-plugins invada-studio-plugins-
ladspa lsp-plugins
```

Estremo Raspberry Pi

Porta Raspberry Pi al limite con questi incredibili progetti.

Dal grazioso pedone
Rob Zwetsloot

Raspberry Pi è un dispositivo molto potente, molto piccolo e molto personalizzabile, e lo abbiamo visto essere utilizzato per tantissime cose nel corso degli anni proprio per questo. Che lo si inserisca in custodie stampate in 3D a forma di per console classiche o semplicemente lo si nasconda come file server domestico, abbiamo trattato molte delle cose interessanti che la community ha realizzato con Raspberry Pi.

Il che solleva la domanda: come si può superare i limiti con un Raspberry Pi? Beh, con 13 anni di progetti e modelli di Raspberry Pi, rimarrete sorpresi di quanto lontano/veloce/elevato/profondo si sia spinto Raspberry Pi. Stiamo portando Raspberry Pi all'estremo.

Hai un utilizzo più estremo di Raspberry Pi? Hai un record aggiornato? Scrivici a magazine@raspberrypi.com

Il più longevo Raspberry Pi

2331 giorni di attività per il Model B

rpimag.co/uptime

"Questo Raspberry Pi è in funzione da dieci anni," afferma l'utente di Reddit KerazyPete, anche con qualche riavvio. Al momento della pubblicazione di questo post, alla fine del 2023, il suo Raspberry Pi 1 Model B (revisione 0002 con 256 MB di RAM, niente meno—una delle prime versioni in produzione) era stato operativo per 2331 giorni consecutivi, cioè da luglio 2017. È il più lungo periodo di attività che abbiamo visto per un Raspberry Pi — e, considerando anche alcune interruzioni di corrente occasionali, sbalzi di tensione e scollegamenti accidentali, è davvero impressionante. Anche i nostri file server e media center sono stati sottoposti ad aggiornamenti hardware.

Puoi facilmente verificare da quanto tempo il tuo Raspberry Pi è acceso aprendo un terminale e digitando `uprecords` - forse rimarrai sorpreso. Anche se consigliamo di aggiornare il sistema operativo del tuo Raspberry Pi ogni volta che viene rilasciato un nuovo sistema operativo, gli aggiornamenti di sicurezza possono essere molto importanti.

▶ Il sistema in questione. Il design originale del Raspberry Pi 1 è piuttosto nostalgico.

^ Il comando `uprecords` mostra da quanto tempo il sistema è in esecuzione

```
pi@node-pi ~ $ uprecords
     #               Uptime | System                                    Boot up
-----------------------------+-----------------------------------------------
->   1   1389 days, 14:22:4 | Linux 4.1.17+            Sun Feb  9 16:58:36 2020
     2   531 days, 01:18:11 | Linux 4.1.17+            Tue Jul 11 22:51:32 2017
     3   371 days, 17:17:30 | Linux 4.1.17+            Sat Feb  2 23:38:37 2019
     4    39 days, 10:36:09 | Linux 4.1.17+            Mon Dec 24 22:16:50 2018
-----------------------------+-----------------------------------------------
NewRec   858 days, 13:04:31 | since                    Sat Jul 24 19:16:46 2021
    up   2331 days, 19:34:3 | since                    Tue Jul 11 22:51:32 2017
  down     0 days, 13:55:14 | since                    Tue Jul 11 22:51:32 2017
   %up               99.975 | since                    Tue Jul 11 22:51:32 2017
pi@node-pi ~ $ sudo tune2fs -l /dev/root | grep 'Filesystem created:'
Filesystem created:       Sat Nov 30 20:00:32 2013
pi@node-pi ~ $ cat /proc/cpuinfo | grep 'Revision'
```

Il Più Grande Raspberry Pi

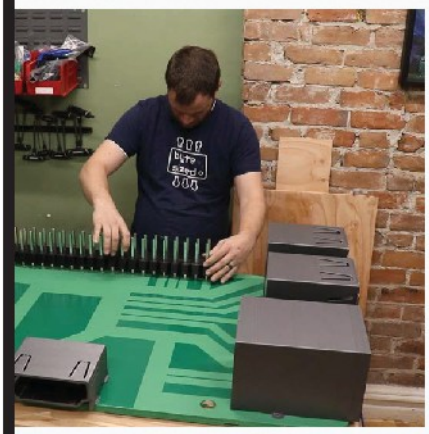
Raspberry Pi 3 gigante funzionante

rpimag.co/largest

Ci sono alcuni enormi computer Raspberry Pi funzionanti. Un Raspberry Pi 3 misura 10X è stato esposto alla Maker Faire Bay Area nel 2016, e Toby Roberts, proprietario di Raspberry Pi, ha costruito un Raspberry Pi 4 misura 6X per una piccola mostra nel centro commerciale dove si trova il negozio ufficiale di Raspberry Pi. Questo Raspberry Pi misura12X funzionante si aggiudica la corona, progettato e stampato in 3D da Zach Hipps.

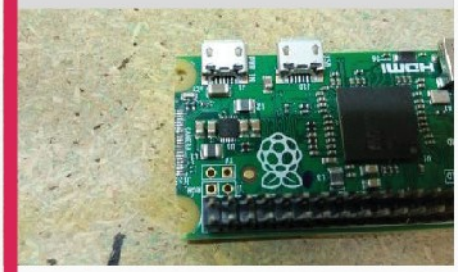
Il PCB è realizzato in compensato, i pin GPIO sono tubi di alluminio e per gli altri componenti di grandi dimensioni sono stati usati oltre 5 kg di filamento in PLA. "Ho collegato il mio Raspberry Pi a tutti i connettori di grandi dimensioni con delle prolunghe", ci ha raccontato Zach quando gli abbiamo parlato qualche anno fa. "Ho collegato un monitor e una tastiera e tutto si è avviato subito!"

▼ Raspberry Pi 3; l'essere umano è per confronto



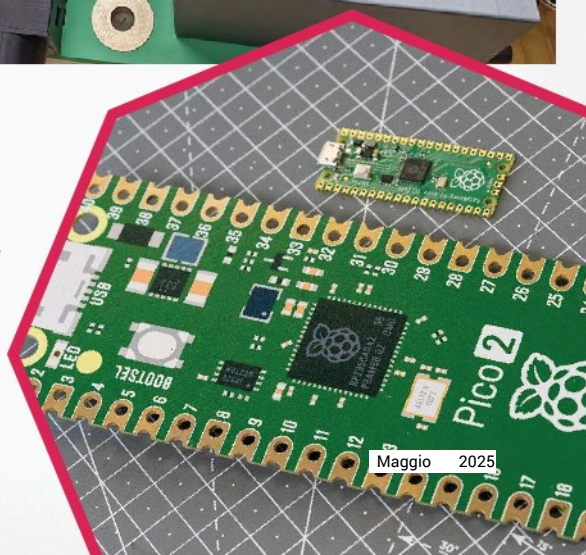
Il più piccolo Raspberry Pi

Sebbene alcune piccole schede RP2040/RP2350 sono tecnicamente i prodotti Raspberry Pi più piccoli, il più piccolo computer Raspberry Pi è il Raspberry Pi Zero. Tuttavia, è possibile ridurlo ancora, fino a 5 mm in meno. Il Raspberry Pi Zero v1.3 ha un connettore per la fotocamera che può essere rimosso dalla scheda e, non essendoci circuiti in quella sezione, è possibile rifilarlo. Consultate il post del forum a riguardo: **rpimag.co/smallest**.



▲ La maggior parte dei dispositivi Raspberry Pi è un po' meno evidente di questo

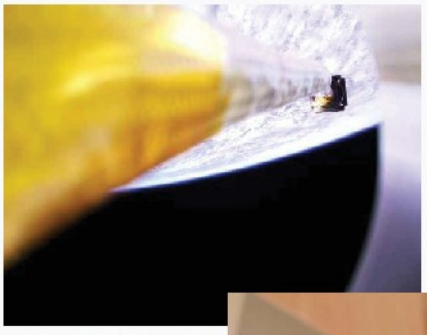
▶ Il Pico Jumbo è un adattatore per Raspberry Pi Pico 2 che lo rende 3,5 volte più grande: **rpimag.co/picojumbo**



Pi nel cielo

Agli albori del Raspberry Pi, e per molti anni a seguire, il velivolo d'alta quota Dave Akerman inviava regolarmente palloni a circa 40 km nella mesosfera (che si trova sopra la stratosfera, per i fan dei Queen) con un Raspberry Pi collegato per scattare foto e acquisire altri dati di telemetria. Nel 2016 ha battuto il record mondiale per l'immagine live più alta inviata da un pallone amatoriale, a 41.837 metri. Probabilmente è la più alta mai raggiunta da un Raspberry Pi senza l'uso di un razzo.

▼ Una foto di una missione di successo



▶ I CubeSat sono molto modesti, ma questo è andato nello spazio con un Raspberry Pi a bordo

Il Raspberry Pi più alto

Highest

GASPACS

428 km sopra la Terra

rpimag.co/highest

È barare andare nello spazio affinché il tuo Raspberry Pi possa rivendicare il record di altitudine per il computer? Non quando ci sono diversi computer Raspberry Pi nello spazio: il più alto attualmente è l'Astro Pi sulla Stazione Spaziale Internazionale. Tuttavia, GASPACS ha battuto il loro record di soli sei chilometri nella sua missione di 117 giorni nel 2022.

Il GASPACS (Get Away Special Passive Attitude Control Satellite) era un CubeSat 1U costruito da studenti della Utah State University per testare l'aerofrenaggio con un "AeroBoom" gonfiabile. Poiché la Terra non è una sfera perfetta, l'altitudine orbitale del CubeSat era di 416 km (258 miglia) al perigeo e di 428 km (267 miglia) all'apogeo.

crediti immagine: ESA

◀ La prima coppia di Astro Pi è stata lanciata per la missione di Tim Peake

Il Raspberry Pi più veloce

Astro Pi

27520 km/h sono più facili senza attrito

astro-pi.org

Sebbene Astro Pi sulla ISS non detenga il record di altitudine per un Raspberry Pi, a causa delle peculiarità della fisica orbitale, è il più veloce. In pratica, avere un'orbita più bassa significa dover andare più veloce per non cadere sulla Terra.

La ISS ha una velocità di 27.520 km/h (17.100 miglia orarie), ovvero 7,67 km/s (4,77 miglia al secondo), ovvero 22,5 volte la velocità del suono a livello del suolo. Questi numeri sono difficili da concettualizzare, ma orbita attorno alla Terra in poco meno di 93 minuti, il che significa che compie 15,5 orbite al giorno. Davvero veloce!

Astro Pi in orbita è dotato di vari sensori grazie a un Sense HAT con sensori di movimento, che vengono utilizzati dagli studenti per eseguire esperimenti tramite codice. A differenza dei GASPACS, questi computer Raspberry Pi sono appositamente rinforzati per il lungo periodo di tempo nello spazio.

crediti immagine: ESA

▶ Un umile Raspberry Pi che vive sulla ISS

Il Raspberry Pi più profondo

Maka Niu

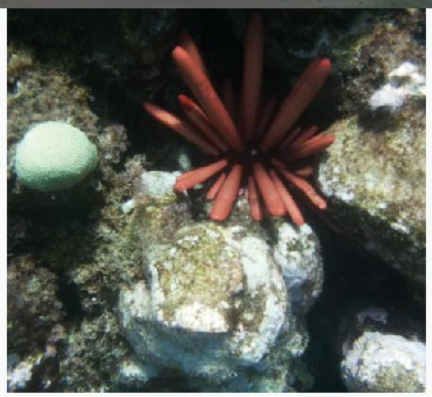
1500 metri sott'acqua – e forse anche 6000 metri – non è un'impresa da poco

rpimag.co/deepest

Scendere sott'acqua in profondità è difficile. Più si scende, più il peso dell'acqua sopra di sé diventa un grosso problema, e i dispositivi artificiali devono resistere alla pressione delle profondità. Ecco perché il tuo orologio potrebbe essere classificato subacqueo solo per 100 metri: poi inizierà a rompersi.

Una discesa di 1500 metri, quasi un miglio sott'acqua, è molto, molto lontana. A questa profondità, in acqua salata, la pressione è di 148 atmosfere, ovvero 148 volte la normale pressione atmosferica a livello del mare.

Maka Niu è quindi un sistema davvero speciale, in grado di contenere in modo sicuro un Raspberry Pi Zero e un Camera Module V2 in un dispositivo a basso costo, aprendo la scienza dei cittadini a più persone e contribuendo a esplorare le profondità marine in gran parte sconosciute.



▲ Potrebbe essere solo un tubo, ma è molto robusto.

◀ Si chiamano ricci di mare perché... Somigliano ai ricci.

Raspberry Pi sommergibile a 300 m di profondità

Pur non potendo raggiungere profondità di 1500 metri, questo capsula pilotabile con Raspberry Pi può raggiungere i 300 metri, un risultato comunque impressionante. Il prezzo di 4250 dollari (3288 sterline) è un po' esagerato, ma almeno dieci volte più economico rispetto ad altri ROV commerciali della sua tipologia. Scopri di più su **bluerobotics.com**.

Il clock più veloce

Raggiungendo il limite di 3,6 GHz

rpimag.co/fastesthz

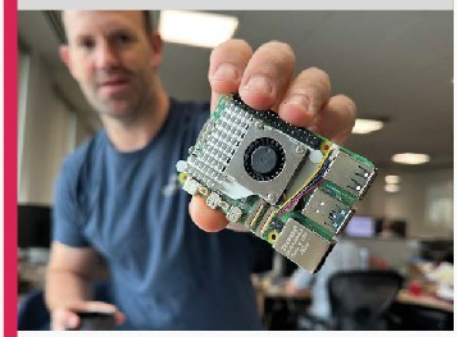
"Raspberry Pi più veloce" può davvero significare due cose, ed è per questo che l'abbiamo goffamente intitolato "Il clocl più veloce". Ogni nuovo modello con numero di Raspberry Pi è leggermente più veloce del precedente, ma la domanda per alcuni è: quanto veloce?

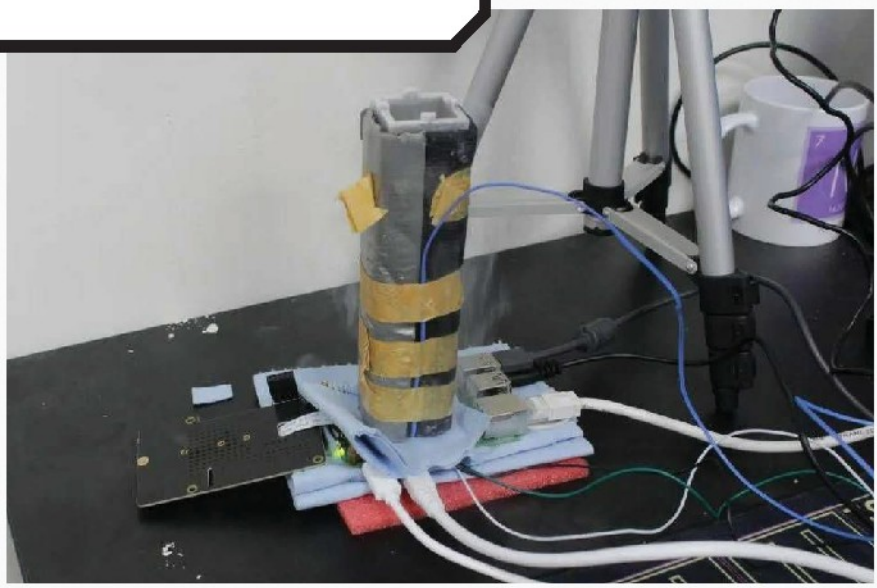
L'overclocking dell'hardware di un computer, ovvero il processo che gli consente di funzionare più velocemente di quanto progettato, aumentando il clock, è una tradizione secolare tra i nerd della tecnologia. L'avvertenza è che surriscalda l'hardware e può danneggiarlo, a lungo termine. Nei normali scenari di overclock, questo problema si risolve con il raffreddamento a liquido o altre soluzioni, ma se si vuole portare un chip ai suoi limiti assoluti, serve qualcosa di veramente freddo: l'azoto liquido. Con un tubo speciale sopra il Raspberry Pi 5, Pieter-Jan Plaisier ha versato azoto liquido direttamente sul chip mentre girava a 3,6 GHz. Può andare più in alto? A quanto pare no: provando a 3,7 GHz il sistema si è bloccato, e non a causa del calore.

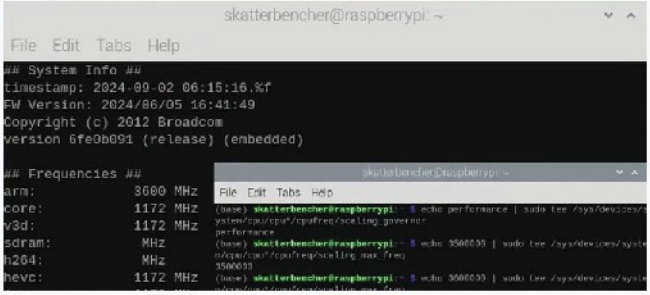
Attenzione!
Azoto
Presta attenzione quando maneggi azoto liquido. Indossa indumenti e occhiali protettivi e segui tutte le istruzioni. Lavora in un'area ben ventilata.
rpimag.co/nitrosafety

Il Raspberry Pi più caldo?

Qual è l'ambiente più caldo in cui opera un Raspberry Pi? Purtroppo, non siamo riusciti a trovare una risposta: alcuni hanno funzionato in deserti che raggiungono regolarmente i 40 °C (104 °F). Il vero trucco è che, con un raffreddamento adeguato, è possibile far funzionare un Raspberry Pi in ambienti con temperature molto elevate, e il chip BCM2712 che gestisce il Raspberry Pi funziona a temperature fino a 80-85 °C (176-185 °F) prima che venga attivato il throttling.







▲ Funziona ma non è molto pratico: è solo un test, dopotutto.

◀ Le prove, sul Terminale, della grande impresa

Raspberry Pi operating

Recently, Compute Module 4 got an extended

▶ Le foto sono state utilizzate per i time-lapse

▼ La stazione ad energia solare scatta foto ad alcuni amici piumati

Il Raspberry Pi più freddo

Monitoraggio dei pinguini di Arribada

Durante gli inverni antartici si arriva a -60°C

rpimag.co/coldest

Inviare qualsiasi attrezzatura in Antartide è complicato: fa molto, molto freddo. Quando la tua tecnologia rimane lì per sbaglio per tre anni e continua a funzionare, è un'impresa notevole. Il progetto di monitoraggio dei pinguini di Arribada ci è riuscito (accidentalmente) non essendo riuscito a ritirare la telecamera alla fine del 2019 e poi, beh, il mondo si è fermato nel 2020 durante l'inizio della pandemia di Covid-19.

Sebbene la temperatura scenda regolarmente sotto i -30 °C (-22 °F), in inverno può raggiungere i -60 °C (-76 °F). L'Antartide è molto estesa e, a seconda del posizionamento della fotocamera, la temperatura varia. Detto questo, dopo tre anni trascorsi nella gelida landa desolata, la fotocamera Arribada è tornata sana e salva con 32.764 foto che i ricercatori hanno potuto esaminare.